From Policy Gradient to Actor-Critic methods
The policy gradient derivation (2/3)

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud
Limits of Algorithm 1

- Needs a large batch of trajectories or suffers from large variance
- The sum of rewards is not much informative
- Computing $R$ from complete trajectories is not the best we can do

\[
\nabla_\theta J(\theta) \sim \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) R(\tau^{(i)})
\]
\[
\approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \left[ \sum_{t=1}^{H} r(s_t^{(i)}, a_t^{(i)}) \right]
\]

* split into two parts

\[
\approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \left[ \sum_{k=t}^{t-1} r(s_k^{(i)}, a_k^{(i)}) + \sum_{k=t}^{H} r(s_k^{(i)}, a_k^{(i)}) \right]
\]

* past rewards do not depend on the current action

\[
\approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \left[ \sum_{k=t}^{H} r(s_k^{(i)}, a_k^{(i)}) \right]
\]

https://www.youtube.com/watch?v=S_gwYj1Q-44  (28’)

2 / 8
Algorithm 2

- Same as Algorithm 1
- But the sum is incomplete, and computed backwards
- Slightly less variance, because it ignores irrelevant rewards
Discounting rewards

\[ \nabla_\theta J(\theta) \sim \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \left[ \sum_{k=t}^{H} r(s_k^{(i)}, a_k^{(i)}) \right] \]

* reduce the variance by discounting the rewards along the trajectory

\[ \sim \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \left[ \sum_{k=t}^{H} \gamma^{k-t} r(s_k^{(i)}, a_k^{(i)}) \right] \]

https://www.youtube.com/watch?v=S_gwYj1Q-44 (39')
Introducing the action-value function

\[ \sum_{k=t}^{H} \gamma^{k-t} r(s_k^{(i)}, a_k^{(i)}) \text{ can be rewritten } Q_{(i)}^{\pi \theta}(s_t^{(i)}, a_t^{(i)}) \]

\[ \nabla_{\theta} J(\theta) \sim \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) Q_{(i)}^{\pi \theta}(s_t^{(i)}, a_t^{(i)}) \]

- It is just rewriting, not a new algorithm
- But suggests that the gradient could be just a function of the local step if we could estimate \( Q_{(i)}^{\pi \theta}(s_t, a_t) \) in one step
Estimating $Q^{\pi \theta}(s, a)$

- Instead of estimating $Q^{\pi \theta}(s, a) = \mathbb{E}_{(i)}[Q^{\pi \theta}_{(i)}(s, a)]$ from Monte Carlo,
- Build a model $\hat{Q}^{\pi \theta}_{\phi}$ of $Q^{\pi \theta}$ through function approximation
- Two approaches:
  - Monte Carlo estimate: Regression against empirical return
    \[
    \phi_{j+1} \rightarrow \arg \min_{\phi_j} \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \sum_{k=t}^{H} \gamma^{k-t} r(s^{(i)}_k, a^{(i)}_k) - \hat{Q}^{\pi \theta}_{\phi_j}(s^{(i)}_t, a^{(i)}_t))^2
    \]
  - Temporal Difference estimate: init $\hat{Q}^{\pi \theta}_{\phi_0}$ and fit using $(s, a, r, s')$ data
    \[
    \phi_{j+1} \rightarrow \min_{\phi_j} \sum_{(s,a,r,s')} ||r + \gamma f(\hat{Q}^{\pi \theta}_{\phi_j}(s', .)) - \hat{Q}^{\pi \theta}_{\phi_j}(s, a)||^2
    \]
  - $f(\hat{Q}^{\pi \theta}_{\phi_j}(s', .)) = \max_a \hat{Q}^{\pi \theta}_{\phi_j}(s', a)$ (Q-learning), $= \hat{Q}^{\pi \theta}_{\phi_j}(s', \pi(\theta)(s'))$ (AC)...
  - May need some regularization to prevent large steps in $\phi$

https://www.youtube.com/watch?v=S_gwYj1Q-44 (36')


Monte Carlo versus Bootstrap approaches

Three options:
- MC direct gradient: Compute the true $Q^\pi_\theta$ over each trajectory
- MC model: Compute a model $\hat{Q}^\pi_\theta$ over rollouts using MC regression, throw it away after each policy gradient step
- Bootstrap: Update a model $\hat{Q}^\pi_\theta$ over samples using TD methods, keep it over policy gradient steps

With bootstrap, update everything from the current state, see next lessons

Next lesson: adding a baseline
Any question?

Send mail to: Olivier.Sigaud@upmc.fr
András Antos, Csaba Szepesvári, and Rémi Munos.
Fitted Q-iteration in continuous action-space mdps.

Martin Riedmiller.
Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method.